恶意代码 SCMP 分类方法框架与风险行为多标签机制

——符合 MECE 原则的分类规范与提升风险揭示度的命名结构改进

肖新光[1]　李晨平[1*]　韩耀光[1]　童志明[1]　李琦[1]

[1]（安天科技集团股份有限公司 哈尔滨 150028）

摘要：

[**目的**] 为响应学术界和工业界对于科学的恶意代码分类方法的需求，

[**方法**] 本研究基于现有工作基础，借鉴了卡巴斯基相对严谨的多段式分类命名的优点，按照强调互斥、完整覆盖、收敛的思路开展，并与“威胁风险行为标签”组合运用，

[**结果**] 形成了一套符合 MECE 原则、分类收敛、兼容工业界事实分类的恶意代码分类框架，

[**结论**] 能够有效支撑安全防御与治理。

关键词：恶意代码　分类方法　威胁行为　风险标签　工程基础

分类号：TP309.5

Scientific Classification of Malware from Practice

Xiao Xinguang[1]　Li Chenping[1]　Han Yaoguang[1]　Tong Zhiming[1]　Li Qi[1]

[1](Antiy Technology Group Co.,Ltd.,Harbin 150028,China)

Abstract：

[Objective] In response to the demand from academia and industry for a scientific classification of malware,

[Methods] this study builds upon existing work and draws insights from Kaspersky's rigorously multi-stage naming methodology, emphasizing the principles of mutual exclusivity, comprehensive coverage, and convergence,combines the use of "threat risk behavior labels",

[Results]. forms a classification framework for malware that conforms to the MECE (Mutually Exclusive, Collectively Exhaustive) principle, achieves convergence, is compatible with real-world industry classifications,

[Conclusions] provides effectively support on security defense and governance.

Keywords： Malware　Classification Method　Threat Behavior　Risk Label Engineering

1　背景

1991 年的 CARO 会议奠定了关于计算机病毒命名的初始行业共识原则，提出了最初的四段式命名法则，业内称之为“CARO 公约”[1]。当时个人计算机系统环境以 DOS 系统为主导，没有广泛的网络链接，磁盘拷贝是软件安装和数据交换的主要介质，所以感染式病毒是彼时主要的威胁形态，其家族和变种总数当时也是在千数量级。而此后伴随着全球信息高速公路建设、互联网应用的快速发展、虚拟和电子资产的不断增值，传统感染式病毒在恶意代码的全貌中已经不再是主

流。网络蠕虫、特洛伊木马等恶意代码的类型先后成为恶意代码风险的主流类型，并不断出现各种新兴的风险和模糊地带，而恶意代码的整体变种数已经膨胀到数以千万计。几乎所有复杂的和规模化的攻击行动都依赖恶意代码的投放和执行。

因此，在安全能力频谱中，恶意代码的发现、检出、精确命名等能力无疑是安全防护的基础支点；在安全防护和运营实践中，反病毒引擎和其他检测机理对恶意代码样本的检出，并触发相关的清除、隔离、拒止等动作，是最基本的安全能力。恶意代码检出需要转化为一个明确的告警提示信息和处置结果反馈：对桌面系统这些信息需要在告警事件窗口进行显示；对企业级别 AV、EPP 或 EDR 的管理界面在事件清单和 TOP 统计中显示；相关日志还需要被 SIEM、XDR 等环节进行联动分析。这就使恶意代码检出事件既要有包括时间戳、对象、是否发现恶意代码、恶意代码名称、处理结果这样的标准结构；与此同时，也需要有规范统一而有一定信息含量的恶意代码命名。网络管理者需要借助相关命名检索关联信息判断风险，相关的 SIEM、XDR 等平台需要借助相关命名进行关联分析和风险优先级排序。

这些工作导致恶意代码命名的质量变得日趋重要，回看 CARO 公约，就会发现其即留下了精确分段命名的遗产，但也留下缺失“分类”概念的遗憾。由于恶意代码家族命名不可避免的是一种带有大量约定俗称的“习惯”而非“范式”的标准，其重点是对应到恶意代码家族的个性化，因此其对关联信息的揭示度往往是不足的。因此在告警的完整的恶意代码名称中，需要一些信息提供恶意代码的共性“属性”，以增加对用户和 IT 运营人员的信息辅助支撑。放到社会层面的安全预警，态势分析和应急响应工作中，监管和应急部门需要超越恶意代码家族和变种的统计“维度”，来分析安全威胁的一些整体趋势和规律。因此，无论是进行信息资产场景中的安全防护和响应，还是进行社会层面的安全治理，也包括以恶意代码作为一种资源进行学术和科研工作，都需要更科学和清晰的分类标准和命名规范，并提供更精确和信息揭示度的恶意代码信息。

对更科学的恶意代码分类方法，学术界和工业界都进行了长期探索，开展了大量的实践。2004 年，卡巴斯基实验室提出了基于“分类树”的恶意软件分类系统，以恶意代码在主机上的行为作为主要依据，并遵循行为风险等级优先原则，实现了分类互斥[2]；赛门铁克、微软、趋势等反病毒厂商也形成了各自的恶意代码分类方式，虽然几乎所有主流厂商都承认“病毒”、“蠕虫”、“木马”三大基础类别的存在，但对三者的定义和类别优先级本身也有一定差异[3][4][5]；美国计算机应急小组领导开展的恶意代码类型枚举工作（CME），以及 MITRE 公司主导的恶意软件属性枚举和特征化工作（MAEC）在推动业内共识、促进安全设备之间的协调方面进行了积极努力[6]。但事实上，面对恶意代码的快速膨胀，试图统一命名带有着不切实际的幻想性。由于缺乏明确的分类依据，安全厂商、团队和机构，相对随意地添加新的分类，而一些新兴的厂商为了创造商业细分，更是进行了一些概念创造，导致各厂商在分类这一本身具有应共性属性的问题上，不断差异化和分叉。以微软为例，截至目前，其基础恶意代码分类已达到 31 种之多。但我们必须指出，恶意代码分类不存在“算法解”。尽管我们看到了诸多的研究文献的尝试，各种基于贝叶斯、近邻计算、图计算来设计所谓“分类算法”，其分类的颗粒度其实对应的都是工业界的对恶意代码的家族概念，而不是分类概念。同时如果没有“工程筛”来进行辅助，几乎所有的基于“熵”、深度学习的算法，都无法应对工业界真实的挑战——即在以百亿级别基础为全集存量，以百万为日增量的样本空间数据基数下算法的可用性。

由于恶意代码对抗的基本运营活动是围绕着工业界的捕获、自动化加人工分析、规则提取、引擎和病毒库升级展开的，对海量样本和分析基础设施的持续建设与掌控，导致工业界一直掌握着事实标准，而且每个主流厂商对恶意代码命名与分类都有自己的经验和内部规范视角。从整体来看，在样本分类命名风格与规范上，已经形成了几个风格鲜明的“流派”。一是以 McAfee、Symantec 等美国主流厂商为代表的“家族派”，其更多继承了 CARO 公约的原始风格，恶意代码多段命名结构中，没有统一的“分类”，只补充了类似 W32 这样的运行环境信息，并添加了少量的关键行为后缀，如@mm。其命名的可理解性、信息揭示度较差。二是以 Microsoft 为代表的“流行派”。由于微软作为安全领域的新生关键力量，没有必须兼容历史命名规范的包袱。因此其恶意代码命名方法主要以流行的威胁类型作为分类标准，主要覆盖 Windows 平台的威胁类型。其视角过于从微软作为操作系统和应用厂商的客户场景和运营情况出发，难以覆盖全量的恶意代码体系。三是以 Kaspersky、Bitdefender 等东欧主流厂商为代表的“行为派”，该派的恶意代码命名方法主要是基于恶意代码的特定行为来支撑分类，并相对严格地遵守“分类前缀”、“环境前缀”、“家族”、“变种号”的四段式命名结构。其结构性和清晰性显著更好。但分类扩展没有明确标准，随意添加，其最多时分类前缀多达近百个。不仅 Virus、Worm、Trojan 这经典的三大基础分类之中出现了类似 P2P-Worm、Email-Worm 等二十余个作为一级前缀的“子类”，而且也不断的产生类似 Backdoor、Rootkit、AdWare、PornWare 这些新的“一级分类前缀”。虽然在卡巴斯基的威胁年报和安全博客统计口径来看，其事实上进行了一定的分类整合与收敛 ，展示出卡巴斯基在根据新的安全威胁和攻击趋势也在尝试调整自身的恶意代码行为分类方法，但始终没有落实到引擎输出的告警信息上。

安天和合作研究者在分类命名上借鉴了卡巴斯基相对严谨的多段式分类命名的优点，但更强调按照互斥、收敛的分类思路开展分类命名，在《安天实验室恶意代码分类标准（2015）》中，安天尝试提出了八种类型的分类法，整体上提出了基于 Trojan 分类吸收所有无主动自我传播能力的高风险样本分类，并特别提出将 TestFile 和 JunkFile 作为两个独立分类。形成了一套对全量样本集合具备互斥性和完整覆盖性的分类框架，并结合了恶意代码核心行为及优先级的命名规范，能够有效覆盖所有恶意代码样本。

基于这些工作基础，本文研究者希望明确提出一套分类规范，并与“威胁风险行为标签”组合运用，其可以和 CARO 公约的命名惯例和事实标准组合运用形成一套新的分类命名逻辑。我们所遵守的工作原则和目标包括：

1、 分类方法符合 MECE(Mutually Exclusive Collectively Exhaustive)原则。按照相互排斥，完全穷尽的标准，该分类方法应能覆盖恶意代码样本，但同时也能将每个样本互斥归属于一个独立的分类空间之中。

2、 分类数量整体收敛，分类间有明确的、带有唯一性条件的分类依据。

3、 兼容目前工业界的事实检出结果，能对多数有严格分类命名规范主流厂商的告警信息实现整合与转化。

4、 能够有效支撑防护场景需求、威胁情报共享、风险预警通报、司法取证和量裁等工作。

2 SCMP 分类法的形成过程

2.1 基础分类：延续经典分类规范

按照复制传播方式这一基础维度，恶意代码的三个最基础分类为：

1、感染式病毒：具有感染宿主属性，借助宿主来进行复制传播；

2、蠕虫：不感染宿主，自身可以进行复制传播；

3、木马：不具备主动感染传播属性，也不进行自我复制。

表 1 恶意代码基本分类表

	感染式病毒	蠕虫	木马
自主感染宿主	是	否	否
自我复制	是	是	否
侵害系统	是	是	是

上述三个分类形成了基于同一维度的基础分类，是恶意代码分类的基础范式。从事实来看上，类似 Emotet、Ryuk 等恶意代码都同时具备既能感染 PE 文件、又能基于网络传播。显然这些样本其同时具备着感染式病毒和蠕虫的双重属性。

因此我们在推荐分类命名的工作中，补充了两条工作原则：

工作原则之一：设定所有分类具有不同的分类优先级别，对有跨分类属性的恶意代码样本，使用分类级别更高的分类。

工作原则之二：在样本命名中引入标签机制，来提升命名的信息输出价值。在一个命名结构中可有多个标签。

当恶意代码有超出本分类特性的其他关键威胁行为时，则使用更细粒度的行为标签进行标识，以便在满足一维分类要求的前提下，提供多元知识结构，避免遗漏信息。

2.2 分类补充和扩展过程

超出 Virus、Worm 和 Trojan 三个基本分类的其他分类扩展过程，本身是对应着本文研究者所面对原有分类不能包含的一些威胁的研究过程。

（1）依托运行的位置扩展出“黑客工具”分类

恶意代码的传统观察视角是收敛于被攻击主机场景的，即围绕恶意代码向被攻击对象的扩散、传播、投放与执行，以及执行后的衍生物。从上世纪末开始，类似 OOB 和 SMBDIE 攻击中的发包器等工具开始广泛被攻击使用，由于其并不影响所运行的主机环境安全，而是影响接收侧主机的系统。并不能被包含在恶意代码的传统分类范围内。但在安全事件响应和取证中，又必须要能发现这些工具。因此，我们以运行位置为一个新的区分度，进行了第一次基础扩展。将运行在攻击者主机上，且本身不具备破坏当前运行主机的完整性、可用性、机密性的能力的工具涵盖在内（否则就应该将其归类为感染式病毒、蠕虫或木马中的一类）。

（2）依托弱侵害风险，扩展出 “灰色软件”分类

传统的恶意代码和黑客工具，通常对应着网络犯罪活动或者出现于带有国家和地区背景的 APT 入侵行为。但在实际的网络行为活动中，还有一些软件和工具用于达成一些弱侵害行为，例如广告软件、色情软件、流氓软件等。而我们提出了用灰色软件涵盖这一类别，其大部分是互联网投放的广告衍生物 ，可能伴随着相关的软件下载而来。有部分杀毒厂商针对类似样本对象使用了“PUA（Potentially Unwanted Application），即潜在有害应用程序。由于分类会直接表现为一级前缀，因此是网络管理者非常关注的事件焦点，灰色软件尽管有很多“行为亚型”，导致部分严格分类的厂商一级前缀严重膨胀，但其整体上处于低风险区，如果任其一级前缀泛滥，就会导致低风险威胁一级前缀在整体一级前缀中形成压倒性占比，导致注意力资源失衡。将其统一收敛到一个大分类，是为

了从告警名称上，整体上降低对用户和网络管理运营人员的干扰和恐慌感，也降低对 SIEM 和 XDR 等环节的合并干扰。

（3）依托非恶意编写目的带来的不确定性风险性，扩展出 “风险软件”分类

恶意代码这一名词的基本要旨其实在于 “恶意”，即其编写者为达成对信息系统的完整性、可用性、机密性等进行侵害的目的。这就使在恶意代码的概念边界中，无法定义以下情况：由合法的机构组织和个人编写，是以支撑实际系统功能和业务为编写目的和实际应用，但有可能被攻击者作为攻击投放物来使用的情况。这就会出现相关软件工具在多数场景环境中是一种正常应用程序，但在少数场景中是一种达成恶意目的工具的情况。基于这种情况，我们认为需要对恶意软件的整个概念范围进行外延，增加 “风险软件(Riskware)”这一类别。例如从 2002 年前后的安全响应处置中，我们遇到了大量的将合法远程网管工具作为远控木马使用的场景，比如，开源网管软件 VNC（远程管理软件）就是典型的风险软件。Riskware 的分类提示有助于网络管理者根据工具是否是自身或使用者部署安装来做出判断。同时，对 Riskware 的分类告警，反病毒软件采取告警提示后等待使用者和管理员处置的方式，避免误杀正常应用。

（4）依托测试机构编写的用户自检有效性样本，扩展 出“测试文件”分类

我们遇到的另一个分类问题是如何对“EICAR”进行分类标识，EICAR 是欧洲反计算机病毒协会(European Institute for Computer Antivirus Research)创建的一种标准测试文件，用于合法地测试和验证防病毒软件和其他安全产品。[7]

通过明确区分实际的恶意代码和测试范例，得出“测试文件”这一独立分类。“测试文件”明确标定了用于检测反病毒引擎和安全防护产品的恶意代码，常见于样本的对抗性测试。

（5）针对无意义样本，扩展出“垃圾文件”分类

我们在多次客户测试场景中遇到了无效测试样本的干扰，这些无效样本往往是客户从网上下载而来，虽然号称样本资源包，但其中的大量文件都并不是样本文件，而是无实际意义的二进制乱码文件。为避免过度陷于对客户样本质量的讨论之中，我们基于“报警和处置掉相关样本不会对系统带来后果或影响且有其他厂商报警”这一原则，对这类样本给予了统一的分类标识。

3 SCMP 分类法分类方法框架

前一章描述的扩展过程初步形成了八个恶意代码分类类别，以图 1 所示的方法框架使这一分类标准成为一个符合 MECE 原则的分类准则。其中，引入了威胁风险、威胁分类、区分依据、分类区隔具体方法和编写者五个维度来构成这一框架。框架整体分类的对象为：迄今为止已经被捕获发现且主流安全产品或检测引擎对其有命名输出的所有对象。

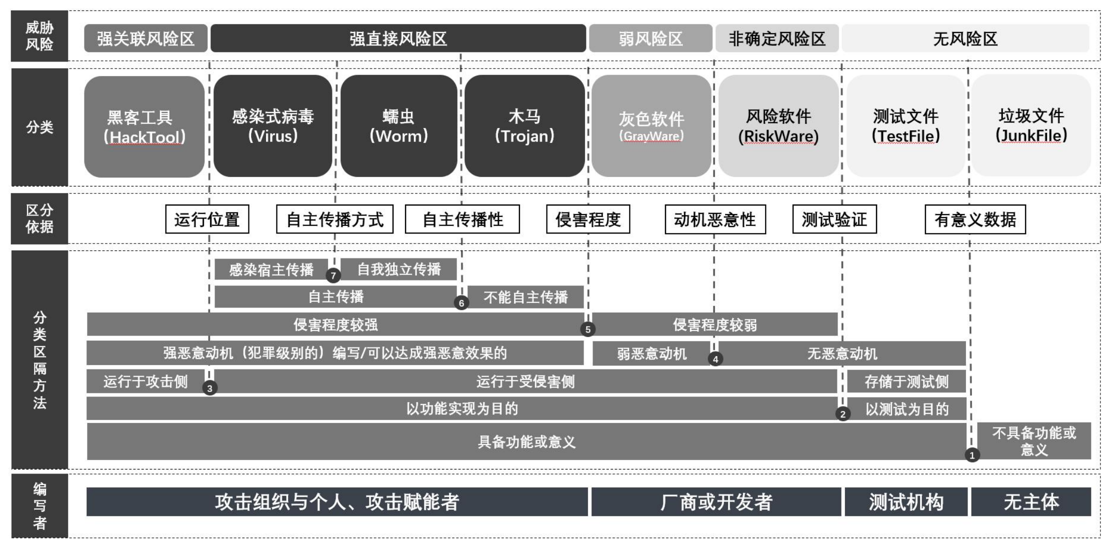


图 1 SCMP 恶意代码分类方法框架

此过程通过七个分类切割点，形成了对恶意代码样本的八个分类：

1. 第一层区分依据是：该样本是否是有意义的数据——将不具备功能或有效意义数据的文件标定为垃圾文件；

2. 第二层区分依据是：该样本的构造目的是为了在用户场景中测试验证还是为了达成功能实现——将以测试验证为目的所形成的样本文件划定为测试文件；

3. 第三层区分依据是：样本所运行的实际位置——将运行于攻击侧、并不运行于受侵害端的样本划定为黑客工具；

4. 第四层区分依据是：动机的恶意性——将无恶意动机、用来执行正常的业务功能和逻辑、但可能被攻击者使用带来风险的样本划定为风险软件；

5. 第五层区分依据是：基于侵害程度的区隔，即犯罪和违法行为的区隔。将侵害程度较弱、侵害后果不构成犯罪但可能违法的样本划分为灰色软件；

6. 第六层区分依据是：在侵害性较强且运行于受害主机的对象中，依托能否实现自我传播进行区隔——将侵害程度较强且不具备自主复制传播能力的样本划分为木马；

7. 第七层区分依据是：在有自主传播能力的样本中，将不依赖感染宿主、可实现自我复制传播的样本划定为蠕虫，将依赖宿主传播的样本划分为感染式病毒。

此分类标准完整覆盖了现有的全量恶意代码样本，满足了分类的互斥性条件，同时也与其他维度建立了较为完整的映射。

例如，对编写者的映射：感染式病毒、蠕虫、木马和黑客工具所对应的编写者为攻击组织、个人和攻击赋能者；灰色软件和风险软件的编写者对应到软件厂商及开发者，测试文件的编写者对应到测试机构，由于垃圾文件成因复杂且没有特定编写者，因此不对应到任何主体。

再如，对威胁风险的映射。因为感染式病毒、蠕虫、木马运行在被侵害主机上，因此其风险程度为“强直接风险区”；黑客工具运行在攻击者或被攻击者控制的终端上，对本身的运行环境无影响，因此其属于“强关联风险区”，而灰色软件的侵害较弱，属于“弱风险区”，风险软件由于在多数场景下为正常软件，在少数情况下被用于攻击，属于“非确定风险区”；而测试文件和垃圾文件既不运行，又不造成实质影响，属于“无风险区”。

对比整个框架切割点的生效顺序以及前置章节中分类的扩展顺序，可以看到，分类切割的顺序基本是扩展分类的逆过程，这也说明了该分类方法本身是经历了威胁对抗和安全运营工作的实战检验的。

4 SCMP的八个基础分类

依据SCMP分类方法框架，恶意代码包含如下八个基础分类，在制定各分类的英文名称时，我们借鉴了帕斯卡命名法(Pascal case)的相关规范。

4.1 感染式病毒（Virus）

定义：感染式病毒是一类以感染宿主的方式完成自我传播的恶意代码。

分类优先级：0（最高）。

说明：感染式病毒的宿主包括不限于：磁盘文件、引导扇区及其它能达成恶意代码自我传播方式的载体。感染式病毒是恶意代码的初始主流形态，其核心特性在于：具有自我复制的特点，且其自我复制需要依赖于宿主。

由于感染宿主是一种破坏了系统基础运行环境和基本运行单元（程序）完整性的行为，因此感染式病毒的分类属性应作为最优先属性，所有恶意代码只要具备主动感染宿主且借助宿主进行传播的属性，无论其有何种其他行为，都应该划分入感染式病毒这一分类。

特别情况和例外清单：由于历史原因，一些并不具备感染式能力也被安全厂商被添加了Virus这一分类前缀，例如DOS时代大量的COM、DOS_MZ和BAT格式的样本。一些厂商的命名前缀中包含Macro Virus，仍归属感染式范畴。与此同时，木马捆绑器（Binder）尽管有类感染行为，但第一其是多数服务于投放过程，而不是在攻击场景内实现持久化，第二，其本身多数并不破坏被捆绑程序本身的完整性，而是添加了独立的文件头。所以其依然被划定到木马类别。

4.2 蠕虫（Worm）

定义：蠕虫是一类不借助宿主即可独立完成自主传播的恶意代码。其自我复制的方式包括基于存储介质和网络方式。

分类优先级：1。

说明：蠕虫通常可以利用系统或软件漏洞、邮件、即时通信、文件共享、社交网络、网络共享或可移动存储设备进行传播扩散，有些蠕虫以网络数据包的形式传播。其核心特性在于：具有自我复制传播的特点且不依赖感染宿主。

特别情况和例外清单：对于通过蠕虫框架进行投放的其他恶意代码部件和组件，如其不是其他已被命名的恶意代码，原则上应作为该种蠕虫的组件或样本。

4.3 特洛伊木马（Trojan）

定义：特洛伊木马是一类以严重侵害运行系统的可用性、完整性、保密性为目的，或运行后能达到同类效果的恶意代码。

分类优先级：2。

说明：尽管木马与黑客工具的编写目的一致，但木马是运行在受害主机中。其核心特性在于运行于受害者环境中，构成强威胁风险。

特别情况和例外清单：安全厂商对于许多具备强风险的威胁有许多分类前缀，例如勒索软件（Ransomware）、挖矿软件（Miner）和后门（Backdoor）等，这些分类不具备自主传播能力，但又具备强恶意动机、具备较强侵害程度。此类威胁可以归为特洛伊木马范畴。

4.4 黑客工具（HackTool）

定义：黑客工具是一类指为达成破坏计算机的可用性、完整性、保密性为目标来编写，但运行在攻击方一侧、起到辅助攻击作用的恶意代码。

说明：尽管黑客工具与木马的编写目的一致，但黑客工具的运行对于当前环境主机不构成相应的威胁风险 。

特别情况和例外清单：远控工具的控制端符合我们对于黑客工具的定义，但由于其生效过程需要和受控端之间在命名上形成映射关系，因此其通常被划分到木马类别，同时用“Backdoor”和“Client”两个行为标签作为修饰。

4.5 灰色软件(Grayware)

定义：灰色软件是一类在受侵害主机上运行、占据被侵害主机的资源、可能带来主机和用户信息泄露，但不足以构成重大风险的软件或插件。

说明：灰色软件和木马的关键区别如下：1）在编写目的上，木马是出于纯粹的侵害目的进行编写，灰色软件在侵害性质的功能之外，也可能包含了部分用户需要的功能；2）在侵害行为上，木马窃取的是用户相对较为关键性的信息，收集环境信息是为了帮助攻击者采取一步的攻击行动，而灰色软件收集信息的目的是为了进行用户画像、广告弹窗以及其他轻量级的牟利变现。3）从组织上，木马通常是由攻击组织、犯罪团体和个人发布的，而灰色软件通常由合法的厂商及开发者进行编写发布；4）从侵害定性上，编写木马属于犯罪行为，编写灰色软件通常属于违法行为。

特别情况和例外清单：安全厂商对于轻量级威胁的许多分类前缀，例如广告软件（Adware）、色情软件（Pornware），实际上都可以划入灰色软件范畴。

4.6 风险软件（Riskware）

定义：风险软件是为了实现某些确定的计算机业务功能而编写的程序，虽然不是为了恶意目的而编写，但有可能在攻击场景下转化为攻击工具。即：其本身的安全风险与“谁安装或投放”、“用于什么目的”相关，而与发布目的无关。

说明：典型风险软件如商用远程工具，如PcAnywhere、VNC等，这些软件在正常使用过程中会在计算机状态栏显示图标，是可以被远程控制者感知到其运行的正常管理工具，但也出现过大量将此类工具作为远控工具来实施攻击活动的案例。

特别情况和例外清单：在攻击事件捕获中，如果明确发现了经攻击者篡改后的工具，则产品会明确的将其标记为特洛伊木马。将这些工具以风险软件的形势进行告警，一定程度上是反病毒企业的权宜之举，以保证既能够发现相关工具被利用的情况，又能避免对用户正常使用的工具产生误报以致造成业务影响或产生法律责任等。在一部分反病毒引擎中，对于此类软件是否报警会设置独立开关。

4.7 测试文件（TestFile）

定义：测试文件是指由测试机构为了让用户能够在自身的场景环境中检测反病毒软件是否能正常工作而发布的公开文档。

说明：测试文件不是指测试机构用来测试安全软件有效性的文件，而是指能够被普通用户使用、通过简单安全的方式在用户自身的系统环境下对安全软件进行自检的文件。原则上，作为测试工具的文件本身应具备涵义，且不应是一个可以执行的程序。

特别情况和例外清单：目前为止，明确符合这一标准的测试文件仅有测试机构发布的eicar，尽管只有一个文件，但其特性可以构成一个独立分支。

4.8 垃圾文件（JunkFile）

定义：垃圾文件是指一些没有实际执行能力和数据意义、但被部分用户或测试机构当作测试样本使用的文件，为了避免此类文件对安全产品的正常工作造成干扰，需要将其作为一个单独的分类。

说明：对正常的可执行程序或数据文件的误报和误选，不应作为告警依据。尽管垃圾文件事实上并不符合恶意代码的定义，但由于其真实存在于反病毒产品的事件告警中，所以需要作为特殊分类，这一分类的存在本质上是由于用户和测试机构的能力不足或误操作而导致反病毒企业必须做出的妥协。

特别情况和例外清单：判断垃圾文件的核心要素是该文件本身是否具有意义。由于反病毒软件对感染式病毒查杀不彻底导致遗留的病毒残体文件，也不应作为垃圾文件，而是应按照恶意代码命名范式并添加 crushed 标签后进行存储。

5 形式化验证

本章通过将恶意代码 SCMP 分类方法描述为一个形式系统，并验证该方法符合 MECE 原则。

5.1 论域

本形式系统所讨论的对象域为：迄今为止已经被捕获发现且主流安全产品或检测引擎对其有命名输出的所有对象。

5.2 符号

表 2 SCMP 分类方法形式系统符号

符号类型	符号格式	含义	示例	示例的语义解释
对象符号	单个小写英文字母	论域内的单个对象	x, y	一个待分类的恶意代码文件对象
断言符号	首字母大写的英文单词/缩略词	断言	Virus (x)	是一个名为 Virus 的、关于对象 x 的断言，其解释为：x 属于“感染式病毒(Virus)”这一分类
量词符号	∀	符合命题的所有对象	∀xF(x)	符合命题 F（x）的所有对象
逻辑符号	->	单向推导	A->B	当断言 A 为真时，B 断言也为真，若 A 则 B。但不能通过 B 反推 A
逻辑符号	<->	双向推导	A<->B	断言 A 与断言 B 同真同假，并且可相互推导
运算符号	¬	非	¬A	对断言 A 的否定
运算符号	&	且	A&B	只有当断言 A 和断言 B 同时为真时，A&B 为真
运算符号	||	或	A||B	断言 A 和断言 B 任一为真时，A||B 为真
运算符号	↓	或非	A↓B↓C↓D	只有当断言 A、B、C、D 均为假时，A↓B↓C

				↓D 为真
标点符号	[ ]	优先运算	¬[A&B||C]	[ ]中的项比[ ]外的项有更高的运算优先级。示例中的逻辑运算顺序依次为&、||、¬
标点符号	/* */	注释	/*sometext*/	注释符号间的文本为注释说明内容

5.3 断言符号及语义

Virus (x)意为：x 属于“感染式病毒(Virus)”这一分类。

Worm(x)意为：x 属于“蠕虫(Worm)”这一分类。

Trojan(x)意为：x 属于“木马(Trojan)”这一分类。

HackTool(x)意为：x 属于“黑客工具(HackTool)”这一分类。

Grayware(x)意为：x 属于“灰色软件(Grayware)”这一分类。

Riskware(x)意为：x 属于“风险软件(Riskware)”这一分类。

TestFile(x)意为：x 属于“测试文件(TestFile)”这一分类。

JunkFile(x)意为：x 属于“垃圾文件(JunkFile)”这一分类。

Nomean(x)意为：x 是不具有功能或有效意义数据的文件。

Test(x)意为：x 是经权威测试机构认证的、以测试验证为目的所形成的样本文件。

Environment(x)意为：x 对其当前所在的运行环境产生侵害。

Male(x)意为：x 是出于恶意动机构造、用来执行恶意功能的文件。

Substantive(x)意为：x 可以构成确定性的、实质性的侵害。

Copy(x)意为：x 具有自我复制的属性。

Inject(x)意为：x 具有主动感染宿主文件的属性。

MECE(E1,E2,E3,,,En)意为：由 E1,E2,E3,,,En 组成的断言集合中，有且只有一个断言为真。

5.4 公理

公理 1：Nomean(x)->Test(x)↓Environment(x)↓Male(x)↓Substantive(x)↓Inject(x)↓Copy(x)

Test(x)||Environment(x)||Male(x)||Substantive(x)||Inject(x)||Copy(x)->¬Nomean(x)

/*无意义的文件不会有任何行为判定点，反之如果文件命中任意一种判定点都说明其是有意义的*/

公理 2：Test(x)->Nomean(x)↓Environment(x)↓Male(x)↓Substantive(x)↓Inject(x)↓Copy(x)

Environment(x)||Male(x)||Substantive(x)||Inject(x)||Copy(x)->¬Test(x)

/*经测试机构认证的 Testfile 不具有实际的行为和侵害后果，如果样本具有了侵害行为特点则说明不是 Testfile*/

公理 3：Substantive(x)->Male(x)

/*以非恶意目的开发的软件无法形成确定性的威胁（因为开发者无法预知以何种方式被利用），有确定性的实质侵害的恶意代码，一定是经过恶意设计或恶意改写的*/

公理 4：Inject(x)||Copy(x)->Environment(x)&Substantive(x)

/*主动感染文件及自我复制已经对当前环境构成了实质性的侵害*/

公理 5：∀x[¬Nomean(x)]&[¬Test(x)]->Environment(x)||Male(x)

/*恶意代码分类工作以现实世界已经发生的威胁为先验经验，正常目的开发且不会对当前环境构成侵害的软件通常不属于反病毒工作的范畴，或者是在白名单中，无论哪种情况都不会使反病毒引擎对其输出恶意代码命名，因此能够进入论域的（除了垃圾文件和测试文件是现实妥协以外），要么确定是以恶意目的开发，要么是非恶意目的开发但是有过被恶意利用形成侵害的先验事实。*/

5.5 推理规则

Nomean(x)<->JunkFile(x)

Test(x)<->TestFile(x)

[¬Nomean(x)]&[¬Test(x)]&[¬Environment(x)]->HackTool(x)

[¬Nomean(x)]&[¬Test(x)]&Environment(x)&[¬Male(x)]->Riskware(x)

[¬Nomean(x)]&[¬Test(x)]&Environment(x)&Male(x)&[¬Substantive(x)]->Grayware(x)

[¬Nomean(x)]&[¬Test(x)]&Environment(x)&Male(x)&Substantive(x)&Inject(x)->Virus(x)

[¬Nomean(x)]&[¬Test(x)]&Environment(x)&Male(x)&Substantive(x)&[¬Inject(x)]&Copy(x)->Worm(x)

[¬Nomean(x)]&[¬Test(x)]&Environment(x)&Male(x)&Substantive(x)&[¬Inject(x)]&[¬Copy(x)]->Trojan(x)

5.6 MECE 原则证明

证明目标：

∀xMECE(Virus(x),Worm(x),Trojan(x),HackTool(x),Grayware(x),Riskware(x),TestFile(x),JunkFile(x))

逻辑用例表：

表 3 逻辑用例表

用例编号	不具有意义	测试专用	恶意开发动机	侵害当前环境	确定实质侵害	主动感染宿主	自我复制	备注	分类结果
1	1	0	0	0	0	0	0	依据公理 1，文件不具有意义时的唯一合法用例	垃圾文件
2	0	1	0	0	0	0	0	依据公理 2，文件是测试专用时的唯一合法用例	测试文件
3	0	0	0	0	0	0	0	依据公理 5 排除用例	
4	0	0	0	0	0	0	1	依据公理 5 排除用例	
5	0	0	0	0	0	1	0	依据公理 5 排除用例	
6	0	0	0	0	0	1	1	依据公理 5 排除用例	

7	0	0	0	0	0	0	0	依据公理 5 排除用例	
8	0	0	0	0	0	0	1	依据公理 5 排除用例	
9	0	0	0	0	0	1	0	依据公理 5 排除用例	
10	0	0	0	0	0	1	1	依据公理 5 排除用例	
11	0	0	0	1	0	0	0	合法用例	风险软件
12	0	0	0	1	0	0	1	依据公理 4 排除用例	
13	0	0	0	1	0	1	0	依据公理 4 排除用例	
14	0	0	0	1	0	1	1	依据公理 4 排除用例	
15	0	0	0	1	1	0	0	依据公理 3 排除用例	
16	0	0	0	1	1	0	1	依据公理 3 排除用例	
17	0	0	0	1	1	1	0	依据公理 3 排除用例	
18	0	0	0	1	1	1	1	依据公理 3 排除用例	
19	0	0	1	0	0	0	0	合法用例	黑客工具
20	0	0	1	0	0	0	1	依据公理 4 排除用例	
21	0	0	1	0	0	1	0	依据公理 4 排除用例	
22	0	0	1	0	0	1	1	依据公理 4 排除用例	
23	0	0	1	0	1	0	0	合法用例	黑客工具
24	0	0	1	0	1	0	1	依据公理 4 排除用例	
25	0	0	1	0	1	1	0	依据公理 4 排除用例	
26	0	0	1	0	1	1	1	依据公理 4 排除用例	
27	0	0	1	1	0	0	0	合法用例	灰色软件
28	0	0	1	1	0	0	1	依据公理 4 排除用例	
29	0	0	1	1	0	1	0	依据公理 4 排除用例	
30	0	0	1	1	0	1	1	依据公理 4 排除用例	
31	0	0	1	1	1	0	0	合法用例	特洛伊木马
32	0	0	1	1	1	0	1	合法用例	蠕虫
33	0	0	1	1	1	1	0	合法用例	感染式病毒
34	0	0	1	1	1	1	1	合法用例	感染式病毒

6　威胁行为风险标签的逻辑和概念

使用基于 MECE 的恶意代码分类原则核心是希望满足分类的覆盖性和互斥性，而在另一层面，由于恶意代码本身功能的复杂性和软件代码本身具有的弹性，导致不可能用有限、收敛的分类集合涵盖恶意代码的所有属性，更无法有效表现出恶意代码的关键威胁信息。一个恶意代码样本可能存在多种威胁行为，所以需要引入一个支持多种表达共存的形态结构，且这种表达有需要足够简洁。显然，将其作为标签是一种更好的选择。

采用多节式结构的恶意代码命名方式在每一分节上都是基于互斥性逻辑。如卡巴斯基的第一节是其恶意代码分类前缀，第二节是环境前缀，第三节是家族名称，后续内容则是变种号和其他修饰符。其每一节都在树形结构中完成对下一分支节点的选择，从而达成相关的互斥性。而我们则在样本命名中统一引入一个风险行为后缀域，在该后缀中，既可以单一输出某个最高等级的风险行为，也可以

输出多个带分隔符的风险行为——前者让使用告警提示信息的相关方关注其最值得响应的行为风险，后者则有更强的信息揭示度。

这一基本工作思路是为了在坚持检测日志的二维化的基础上，保证相关的样本命名和行为信息依然以单一字符串的方式来输出，同时也为细粒度的事件和知识信息查询中把样本的命名转化为多维的数据结构提供了基础。

在建立行为标签时的基本考虑要素包括：

1、涵盖不适宜作为分类，但当下比较流行、有较大风险活用户高度关注的安全风险，例如勒索、漏洞利用、欺诈、内核伪装等；

2、能够用以映射和消化主流安全软件，在原有的、不符合 MECE 的标准体系下所输出的小类别或一级前缀；

3、涵盖恶意代码关键行为的几个维度，如：传播方式、攻击目的及对象、攻击技巧、隐蔽方式等。

安天基于这套行为标签的引导，在静态分析、动态沙箱的开发过程中形成了对样本行为的动、静态的判断机制，使这些行为标签可以在自动化分析过程中产生，并且发给第三方使用。

对象行为标签的引入，不仅弥补了基于收敛分类的方法信息揭示不足的问题，也能够基本上把所有的主流厂商的有效恶意代码命名转化为基于基础分类、运行环境、家族名、变种号和行为后缀的对应命名；同时，针对同一样本，不同引擎能够供给不同对象信息时，也能将这些信息转化为命名中的有效信息，可以支撑应急响应组织的尝试统一告警格式与风格的实践。

7 对现有分类的合并吸收

7.1 合并分类吸收表

SCMP 通过八个分类，可以完整吸收卡巴斯基等精准分类命名厂商的现有分类和一级前缀，具体表格如下。

表 4 SCMP 分类吸收表

SCMP 分类类别	吸收的厂商分类和一级前缀
感染式病毒（Virus）	• Kaspersky：感染式病毒（Virus） • Mirosoft：宏病毒（Macro virus）
蠕虫（Worm）	• Kaspersky：蠕虫（Worm）电子邮件蠕虫（Email-Worm）、即时消息蠕虫（IM-Worm）、网络蠕虫（Net-Worm）、对点文件共享网络（P2P-Worm）
特洛伊木马（Trojan）	• Kaspersky：特洛伊木马（Trojan）、勒索型木马（Trojan-Ransom）、后门（Backdoor）、内核套件（Trojan-Rootkit）、网银木马（Trojan-Banker）、流量劫持木马（Trojan-Clicker）、下载者木马（Trojan-Downloader）、释放器木马（Trojan-Dropper）、包裹炸弹（Trojan-ArcBomb）、间谍木马（Trojan-Spy）、拒绝服务攻击木马（Trojan-DDoS）、僵尸网络木马（Trojan-Botnet）、挖矿木马（Trojan-Miner）、代理型木马（Trojan-Proxy）、拨号型木马（Trojan-Dailer）、键盘记录器（Trojan-Keylogger）、盗号木马（Trojan-PWS）等、仿冒杀软（Trojan-FakeAV） • Bitdefender：特洛伊木马（Trojan）、漏洞利用（Exploit）、键盘记录器（Keylogger）、后门（Backdoor）、下载器（Downloader）

	• McAfee：特洛伊木马（Trojan）、密码窃取（PWS） • Symantec：特洛伊木马（Trojan）、后门（Backdoor）、挖矿（Miner）、下载器（Downloader）、勒索软件（Ransom） • Mirosoft：特洛伊木马（Trojan）、勒索软件（Ransomware）、漏洞利用（Exploit）、后门（Backdoor）、下载器（Downloader）、释放器（Dropper）流氓安全软件（Rogue security software）、密码窃取器（Password stealer）、流量劫持木马（Trojan-Clicker）、远程控制程序（Command and Control）
黑客工具（HackTool）	• Kaspersky:黑客工具（Hacktool）、生成器(Constructor)、病毒工具（VirTool） • Mirosoft：黑客工具（Hacktool）、混淆器（Obfuscator）、病毒工具（VirTool）
灰色软件（Grayware）	• Kaspersky:广告件（Adware）、色情软件（Pornware） • Bitdefender：间谍软件(Spyware)、色情软件（Porn）、广告件（Adware） • McAfee：潜在有害程序（PUP） • Mirosoft：潜在有害应用程序（PUA） • AVG：潜在有害程序（PUP）
风险软件（Riskware）	• Kaspersky：RemoteAdmin（远程管理软件）、监控程序（Monitor）、网络工具（NetTool）、风险工具（RiskTool） • Mirosoft：远程管理软件（RemoteAccess）
测试文件（TestFile）	• Kaspersky：EICAR 测试文件（EICAR-Test-File） • Bitdefender：EICAR 测试文件（EICAR-Test-File）
垃圾文件（JunkFile）	• 其他厂商未覆盖

7.2 一些典型分类被吸收情况的说明与相关认知误区的澄清

SCMP 由于大量减少了恶意代码分类数量，我们需要针对一些典型的恶意代码的相关认知误区进行澄清。

当前关于恶意代码的基本分类存在一些误区，究其原因，既有对恶意代码认知的不充分，也有翻译问题以及历史工程因素。下文将介绍几个常见的分类误区，并阐述其为何不符合科学性。

（1）Backdoor（后门程序）

在计算机病毒分类命名体系中，“Backdoor”一词首见于 2000 年前后，最早被标记为 Backdoor 的恶意代码是由死牛崇拜（The Cult of the Dead Cow）组织发布的远程控制工具，后续与之相类似的远程控制的恶意代码如“网络神偷”、“冰河”等，都被主流杀毒厂商标记为后门。可见，“Backdoor”这一分类前缀实际并不是指在软件中存在的代码缺陷，而是由于最早的控制工具使用的是 Back Orifice，使其成为了一个能够直接访问整个系统的后门。随着后来相关的恶意代码穿透内网、远程管理技术的升级，相关样本依然被命名为后门，“后门程序”这一定义的实际内涵是指那些具有远程控制能力的木马。

因此，在 SCMP 分类命名法中，此类恶意代码统一被归类为“木马”，而“Backdoor”则作为最高风险级别之一的标签出现，并不影响信息揭示度。

（2）Spyware（间谍软件）

在国内外的一些文献中，“Spyware”被解读为：“具有间谍活动能力的恶意代码”，事实上这种解读是望文生义的，并不符合该词语的原始内涵。反病毒工作者最早提出“Spyware”的时代背景为互联网客户端的大规模盛行，一些插件在未经用户同意的情况下，安装到用户的开机目录下，成为隐蔽运行的默认插件。“Spy”传达的是软件程序的“静默进入”这一安装特点，而非 “信息窃取”这种威胁行为，后期对间谍软件的解读，事实上都是一种误读。

因此，在 SCMP 分类命名法中，此类恶意代码被归类为“Grayware”，而“Spy-Install”则作为一种恶意代码的行为标签，并不影响信息揭示度。对于明确有敏感和关键信息窃取行为的恶意代码，可酌情划入“Trojan”类别。

（3）Botnet（僵尸网络）

在一部分后发的检测软件的命名中出现了类似 Botnet 这样的分类，但僵尸网络并不是一个样本概念，一定程度上包含了样本运用和组织方式，如果将其作为一种类别，就是恶意代码的分类体系上引入了一个新的分类维度，这显然违背了恶意代码分类的基本原则，事实上是对分类标准的干扰。

从病毒样本的角度来看，僵尸网络运行在受害端的程序和具有远控能力的木马的机理是一致的，差异点仅在于前者按指令进行自动化控制的能力相对更强。

因此，在 SCMP 分类命名法中，此类恶意代码统一被归类为“Trojan”，而“Botnet”则作为高风险级别的标签出现，并不影响信息揭示度。

（4）Adware（广告软件）、Pornware（色情软件）、Rogue（流氓软件）

广告软件和色情软件也曾被部分杀毒软件当成独立分类，但由于基于互联网的灰、黑产其实是联合运营体系；在广告软件所弹窗的内容中也往往有大量的色情文件，导致这两类很难形成明确的区分度。二者的本质都是通过特定内容吸引用户的注意力，引导用户点击、下载，方便进一步开展后续的非法信息收集等侵害活动。流氓软件是未经用户明确授权或知情的情况下，在计算机或移动设备上安装。流氓软件、广告软件和色情软件通常都是通过捆绑在其他软件、下载器或共享软件中，或者通过欺骗性的下载链接、钓鱼邮件等方式传播。流氓软件和广告软件这两种软件虽然都可能对用户造成不便，但它们一般不会对系统或数据造成直接破坏或危害。

因此，在 SCMP 分类命名法中，此类恶意代码统一被归类为“Grayware”，而“Ad、Porn”则作为高风险级别的标签出现，并不影响信息揭示度。

（5）Ransomware（勒索软件）

毫无疑问，勒索软件是当前最严重的安全威胁，对流行的勒索软件进行告警和处置非常重要，但这并不意味着“勒索软件”适合成为一个独立的恶意代码基础分类。因为无论勒索软件采用的是何种工作机理和模式，在本质上都没有逃逸出对“木马”的定义。

因此，在 SCMP 分类命名法中，此类恶意代码统一被归类为“Trojan”，但“Ransom”则作为最高风险级别的行为标签出现，并不影响信息揭示度。

（6）关于构造器（Constructor）和混淆器（Obfuscator）

病毒生产机本身是一个比较古老的概念，其整体上还是在 DOS 下可执行文件（如 COM 文件）没有文件格式规范、DOSMZ 也没有严格的格式验证的背景下，基于相关的功能模块组合及混淆操作来生成病毒样本。与此同时，由于感染式病毒本身代码段并不是独立的可执行程序，因此对编写者来说，想达成初始的感染，就要先构造宿主，因此才出现了构造器的概念。

可见，构造器是一个DOS时代的早期概念，引入当前环境场景中的意义已经不是很大。与之类似，一些模块化木马有其定制加工的配置界面，但其配置器本身可以按照前文的定义划分到“黑客工具”类别，也可以基于“与投放物配套”的原则，划分入木马分类。

混淆器概念是在DOS时代变形引擎的时代出现，其核心机理是实现变形引擎与病毒载荷的融合，从而使没有进行相关的混淆、变形的样本，在结合了变形引擎之后具有变形能力。在恶意代码形态从感染式病毒主导演变为独立式、甚至固件式为主导的情况下，事实上只有样本加壳概念，而不再有作为恶意代码的混淆器的概念。

由于大部分加壳工具服务于版权保护场景，只有在特定攻击场景中，才会被作为逃逸检测的行为特征，如果简草率地报警壳，则会对正常应用软件产生误报。目前反病毒厂商会对一些仅在攻击场景中使用的地下壳进行报警，但其告警往往是以可命名的方式的事件提示信息出现（而非告警信息），以示谨慎。从整体性上来看，所有主流杀毒产品不会对壳进行单独告警，因此，已经不适宜将混淆器作为一个恶意代码的分类类别。

8 最终的效果和实践应用

我们将整体应用“类别”/“环境前缀”、“家族名”、“变种号”、[行为标签]作为恶意代码分类框架。以如下样本信息命名为例：Trojan/Win32.Akira[Ransom]。在实际的安全业务中，我们既能明确地看到这是一个运行于Windows32位平台的特洛伊木马，也能看到其核心风险行为是进行勒索攻击。而在自动化预警通报中，基于对这个字符串的结构解析，则可以转化为“类似勒索风险预警，警惕Akira木马”的安全通告内容。

8.1 科学分类方式对网络安全管理运营的意义

网络安全管理运营工作需要科学清晰的恶意代码分类标准和命名规范。恶意代码样本是海量的、其分布是长尾的，绝大多数用户已经不具备自己单独分析恶意代码的能力，在精准识别和细粒度处理方面必须依赖于安全产品。安全产品对恶意代码精准命名和明确区分恶意代码分类的能力，与网络安全管理运营者需要考虑的运营问题以及如何构建对应的基础运行处置流程之间存在关联。

SCMP恶意代码分类方式能够提供更准确和具有更高信息揭示度的恶意代码信息，支撑网络安全管理运营者进行威胁识别、风险评估、安全策略制定、安全事件响应和处置流程构建，将有限精力聚焦于几个确定性的处理流程，从而提升运营处置效果和效率。

基于分类标准的几个区分度来看，如果发现感染式病毒（Virus）、蠕虫（Worm）两种具有主动传播能力的告警事件，说明用户可能缺失基本的安全合规能力，或者没有遵守基本的安全基线；如果发现特洛伊木马（Trojan）告警事件，则需要针对性分析处置；如果发现黑客工具（HackTool），说明主机可能成为跳板或被用来发起横向移动、窃取敏感信息或发起进一步的攻击，但也可能是内部的红队（Red Team）工具；如果发现灰色软件（Grayware），说明存在弱信息泄露侵害，或治理能力需要提升，但亦可选择搁置；如果发现风险软件（Riskware），则需要排查是否是网管人员主动安装使用，如果是则需要添加白名单，如果不是则可能转入到Trojan的处置流程；网络管理者可以用垃圾文件（JunkFile）告警判断样本集合的质量，同时JunkFile也可以用来降低用户的恐慌感；而测试文件（TestFile），除非用户主动部署，否则一般不会出现，如果在用户系统内

发现了 TestFile，则需要排查是不是攻击者构造的伪装测试文件。这样再辅以核心行为标签，就能把海量恶意代码的处置收敛到几个确定性的流程中。

表 5 恶意代码类别所关联的运营动作

	基线检查与加固	问询	样本分析	关注后续行为	猎杀清除	配置更新	忽略
病毒（Virus）	*	NA	√	NA	*	√	-
蠕虫（Worm）	*	NA	√	NA	*	√	-
特洛伊木马（Trojan）	√	NA	√	√	*	√	-
黑客工具（HackTool）	√	*	√	√	√	√	√
灰色软件（Grayware）	√	NA	NA	NA	√	√	√
风险软件（Riskware）	√	*	NA	NA	NA	√	√
测试文件（TestFile）	NA	*	√	NA	NA	NA	√
垃圾文件（JunkFile）	NA	NA	NA	NA	√	NA	√
*建议采取 √酌情采取 -不建议采取 NA 不相关							

2.2 实践应用

上述的分类命名的基本原则已经在安天 AVL SDK 反病毒引擎客户中实际工程化应用实践，有近 100 家合作伙伴使用了我们的检测引擎，根据我们不完全统计，超过了 130 万台网络设备和网络安全设备、超过 200 万个 PC 和云节点和超过 30 亿部手机和智能终端使用了这一反病毒引擎。

与此同时，我们依托这一命名结构的严谨互斥性，基于大量恶意代码数据分析据，并在大模型的辅助下构建计算机病毒分类命名知识百科，截至 2023 年底，实现收录病毒家族词条 53,000 余条，分类命名对已知恶意代码家族覆盖率达到 100%，目前进入日增量家族维护状态。

除了我们自身的安全实践外，基于这套分类命名结构基本能无损吸收其他主流安全厂商的所有分类和命名信息，其可以支撑应急响应组织的尝试统一告警格式与风格的实践。

当然面对持续膨胀的安全威胁和不断丰富的攻击样式。防御者难免陷入艰难和迷惘。但代码对抗依然是网络空间对抗最基本的模式；恶意代码依然是绝大多数网络攻击活动中的攻击武器，能有效识别并遏制恶意代码对防御者来说就更为关键。可以说，恶意代码的检测已经成为网络空间对抗中的“敌我识别”能力，而我们所做的是尽可能的让我们的精确分类检测和标识能力尽可能多覆盖的攻击者的“攻击武器”。

参考文献：

[1]Vesselin Vladimirov Bontchev.Current Status of the caro malware Naming Scheme[EB/OL].[2024-04-29]. https://bontchev.nlcv.bas.bg/papers/naming.html#1.1.4.%20Lack%20of%20Reliable%20Means%20for%20Automatic%20Malware%20Identification

[2]Kaspersky Lab[EB/OL].(2013-12-10)[2024-04-29].https://encyclopedia.kaspersky.com/knowledge/rules-for-naming/.

[3] Symantec[EB/OL].[2024-04-29]https://www.broadcom.com/support/security-center
[4]Microsoft[EB/OL].(2024-04-22)[2024-04-29].https://learn.microsoft.com/en-us/microsoft-365/security/intelligence/malware-naming?view=o365-worldwide.
[5]Trend Micro[EB/OL].(2019-12-30)[2024-04-29].
https://success.trendmicro.com/dcx/s/solution/1119738-new-threat-detection-naming-scheme-in-trend-micro?language=en_US.
[6]The MITRE Corporation.[EB/OL].[2024-04-29].https://maecproject.github.io/.
[7]European Institute for Computer Antivirus
Research[EB/OL].[2024-04-29].http://www.eicar.org/.

(通讯作者: 李晨平 E-mail:lichenping@antiy.cn)

作者贡献声明：

肖新光：提出研究思路，设计研究方案；
童志明，韩耀光：进行实验；
韩耀光，童志明：采集、清洗和分析数据；
肖新光，李晨平，韩耀光，童志明：论文起草；
李琦：绘制插图；
李晨平：论文最终版本修订。